I was not able to run this exact test because my virtual machine does not have 6.4 GB of space to spare and I also do not want to wear out my SSD performing a lot of trivial writes. However, I did run similar trials of my own on smaller files.

I did not observe any significant speedup in my final implementation.

I was not able to implement a good multi-threaded server. I spent a huge amount of time on this assignment and it is very frustrating to not get it after all of it. My final result has a lock over my entire handle_task() function, which effectively makes my server single-threaded. Even running with a single thread, I run into errors where recv() gets hung waiting for data.

I have identified three critical areas where potential errors may occur: process_http_request(), construct_http_response(), and write_log(). Process and request present a possible hazard because two subsequent requests could attempt to modify the same file at the same time. Additionally, write_log() must be given to one thread at a time because it will always write to the same log file. Reading a request is safe because each thread has its own httpObject struct that it can read to.

I believe it would be possible to improve this by only locking the file that a thread is currently working on. This would ensure that two threads could process requests to separate files. I also believe I could change how write_log() is implemented so that it takes an offset that is equal to how many bytes are or will be present in the log file. Then each thread only writes to the space in the log file that it has been given.

Another possible bottleneck for this system would be the disk read/write speed. I would imagine that using a slow hard disk would bottleneck this server regardless of how many threads are running concurrently. The only way to improve this bottleneck would be to get a faster hard drive or SSD.

In real life, we would not log the entire contents of files for several reasons. First, we do not need to see the data that was transferred in a log, we can just look in the file that was written by the server assuming we have correctly implemented the write. Second, the server would require way more space because we would have a HUGE log file with the contents of every single file that has been transferred. In a real world implementation, this would quickly become very large. Third, a log is primarily used to just see the metadata and relevant information about your servers status for each request, the transferred data is not relevant.